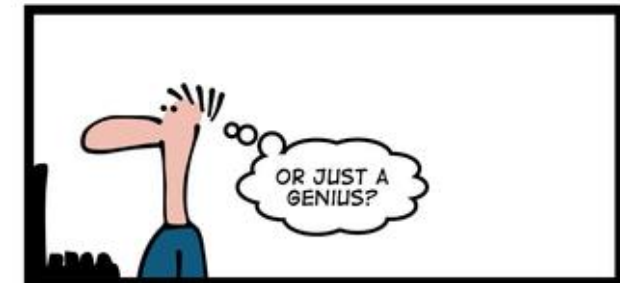
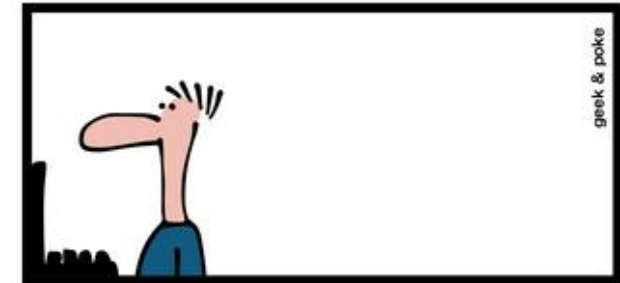


CSc 337

LECTURE 25: COOKIES

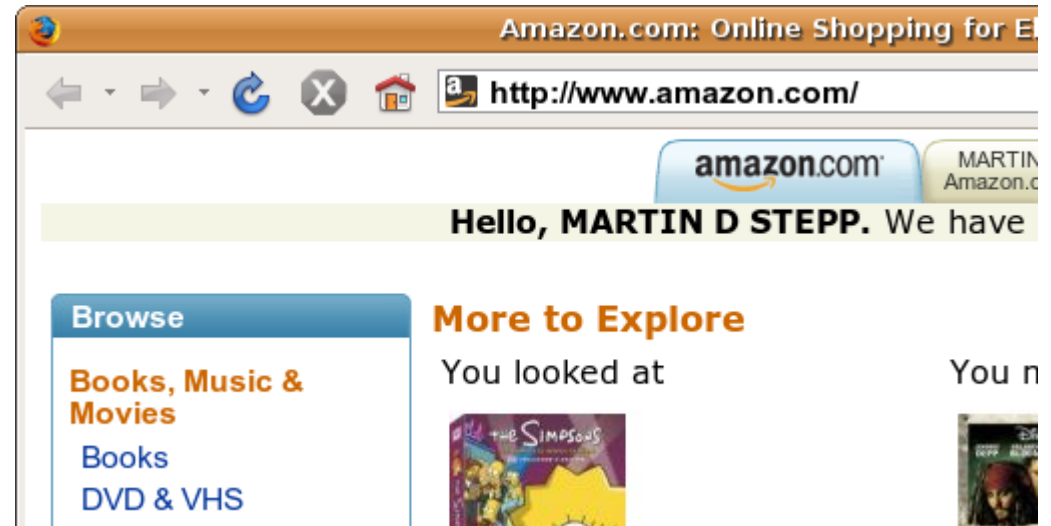


YESTERDAYS REGEX

Stateful client/server interaction

Sites like amazon.com seem to "know who I am." How do they do this? How does a client uniquely identify itself to a server, and how does the server provide specific content to each client?

- HTTP is a **stateless** protocol; it simply allows a browser to request a single document from a web server
- today we'll learn about pieces of data called **cookies** used to work around this problem, which are used as the basis of higher-level **sessions** between clients and servers



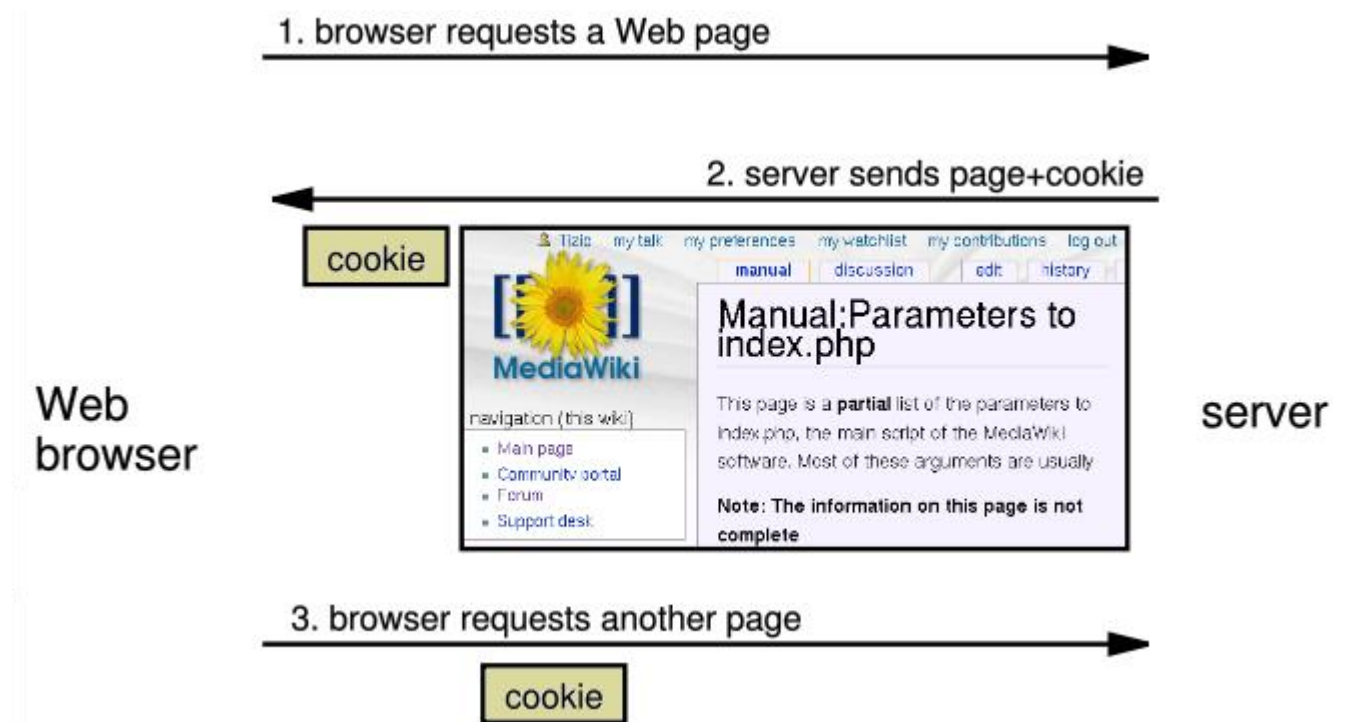
What is a cookie?

- cookie: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests
- cookies have many uses:
 - authentication
 - user tracking
 - maintaining user preferences, shopping carts, etc.
- a cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request



How cookies are sent

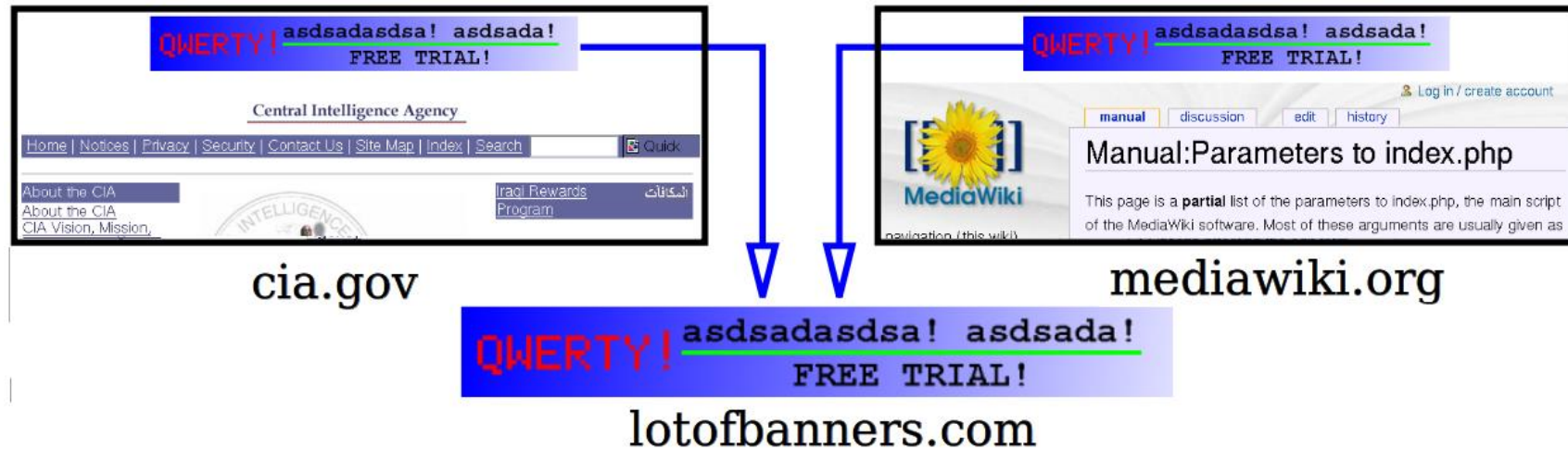
- when the browser requests a page, the server may send back a cookie(s) with it
- if your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests
- alternate model: client-side JavaScript code can set/get cookies



Myths about cookies

- Myths:
 - Cookies are like worms/viruses and can erase data from the user's hard disk.
 - Cookies are a form of spyware and can steal your personal information.
 - Cookies generate popups and spam.
 - Cookies are only used for advertising.
- Facts:
 - Cookies are only data, not program code.
 - Cookies cannot erase or read information from the user's computer.
 - Cookies are usually anonymous (do not contain personal information).
 - Cookies CAN be used to track your viewing habits on a particular site.

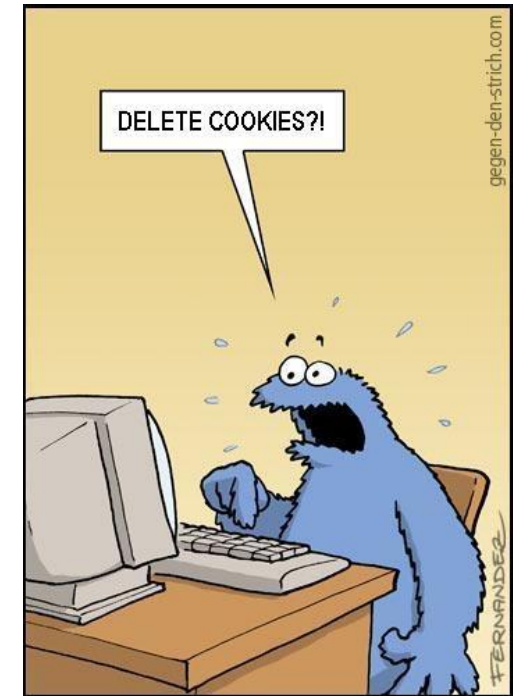
A "tracking cookie"



- an advertising company can put a cookie on your machine when you visit one site, and see it when you visit another site that also uses that advertising company
- therefore they can tell that the same person (you) visited both sites
- can be thwarted by telling your browser not to accept "third-party cookies"

Where are the cookies on my computer?

- IE: *HomeDirectory*\Cookies
 - e.g. C:\Documents and Settings\jsmith\Cookies
 - each is stored as a .txt file similar to the site's domain name
- Chrome:
 - C:\Users*username*\AppData\Local\Google\Chrome\User Data\Default
- Firefox: *HomeDirectory*\.mozilla\firefox\???.default\cookies.txt
 - view cookies in Firefox preferences: Privacy, Show Cookies...



How long does a cookie exist?

- **session cookie** : the default type; a temporary cookie that is stored only in the browser's memory
 - when the browser is closed, temporary cookies will be erased
 - can not be used for tracking long-term information
 - safer, because no programs other than the browser can access them
- **persistent cookie** : one that is stored in a file on the browser's computer
 - can track long-term information
 - potentially less secure, because users (or programs they run) can open cookie files, see/change the cookie values, etc.

Setting Cookies

You will need to install cookie-parser in order to use cookies in NodeJS:

```
npm install cookie-parser
```

In order to use cookie-parser in your code you will need to include the following lines:

```
const cookieParser = require('cookie-parser');  
app.use(cookieParser());
```

Setting a cookie in NodeJS

```
res.cookie(cookie_name, cookie_value)
```

```
res.cookie('username', 'allison')
```

- you can set multiple cookies (20-50) per user, each up to 3-4K bytes
- by default, the cookie expires when browser is closed (a "session cookie")
- you can check whether a cookie has been sent by typing **document.cookie** into the browser console

Retrieving information from a cookie

```
req.cookies          // retrieve value of the cookies  
var name = req.cookies.name;
```

Cookies can be retrieved from the request

Expiration / persistent cookies

```
res.cookie(name , 'value', {maxAge : 10000});  
  
var expireTime = 60*60*24*7; // 1 week from now  
res.cookie("CouponNumber", "389752", {maxAge : expireTime});  
res.cookie("CouponValue", "100.00", {maxAge : expireTime});
```

- to set a persistent cookie, pass a third parameter for when it should expire
 - time is in milliseconds
- indicated as an integer representing a number of seconds, often relative to current date
- if no expiration passed, cookie is a session cookie; expires when browser is closed


Deleting a cookie

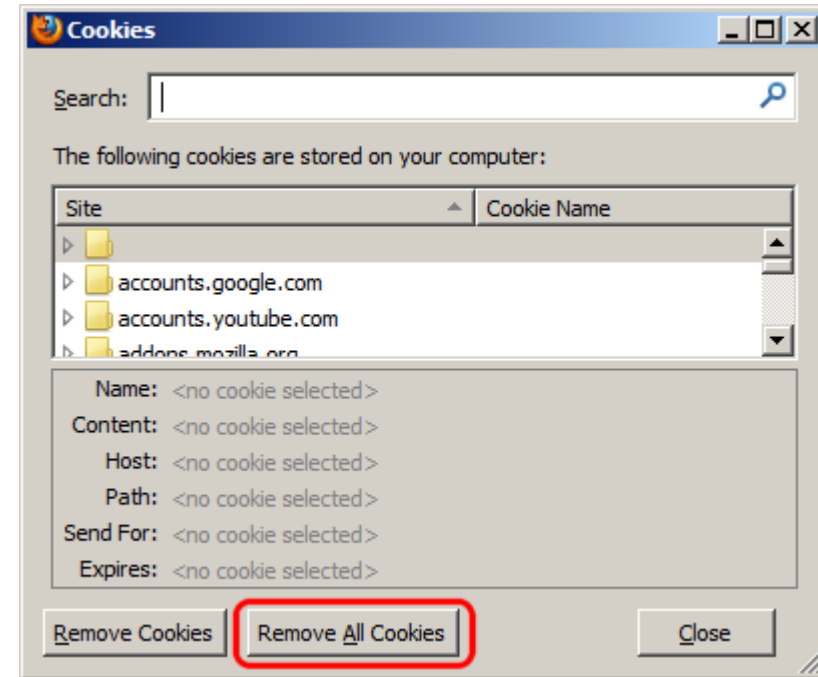
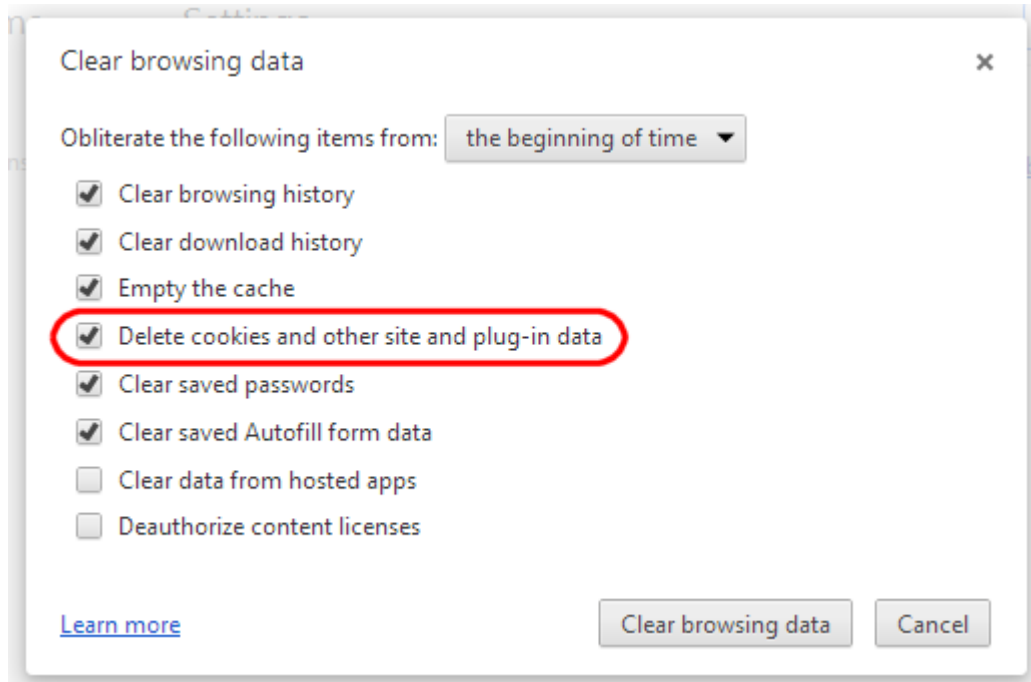
```
res.clearCookie(cookie_name);
```

```
res.clearCookie('name');
```

- takes the name of the cookie to delete as a parameter
- remember that the cookie will also be deleted automatically when it expires, or can be deleted manually by the user by clearing their browser cookies

Clearing cookies in your browser

- **Chrome:** Wrench  → History → Clear all browsing data...
- **Firefox:** Firefox menu → Options → Privacy → Show Cookies... → Remove (All) Cookies



Cookie and Session error fix

Some students have trouble getting cookies and sessions to work on their machines. If you can't get them to work, try the following:

- Use Firefox not Chrome
- change your fetch call to: `fetch(url, {method: "GET", credentials: "include"})`
- Include `httpOnly : false` when you set your cookie
 - example: `res.cookie("luckynum", number, {maxAge : 10000, httpOnly : false});`
- Access your html page from `http://localhost:3000/page_name.html`
 - to do this you will need to move your `.html` files to a folder called `public` located in the folder that your service is stored in.